Feature Fusion Network for Skeleton-based Action Recognition

Wei Guo[1], Haonan Ma[1], Zikai Li[1], and Jianwen Chen[1]

[1] South China Normal University, Guangzhou 510006, P. R. China

Abstract: With the increasing demand for intelligence in human-computer interaction, security monitoring, intelligent nursing, and sports analysis, the development of human skeletal behavior recognition technology has attracted more attention. However, current methods based on human skeleton recognition encounter a balance issue between model complexity and accuracy, and struggle to comprehensively extract the required features. To address these challenges, this study proposes a Feature Fusion Network based on ST-GCN as the backbone network, which achieves comprehensive and detailed feature extraction through multiple feature fusion operations within the network. The parameter count of FFN is only 3.35 million. It achieves accuracies of 94.24% and 98.30% on the 2D skeletal data of the NTU RGB+D 60 dataset using the cross-subject and cross-view partition criteria, respectively. On the NTU RGB+D 120 dataset, it achieves accuracies of 87.31% and 90.95% using the cross-subject and cross-setup partition criteria, respectively, representing a state-of-the-art performance in the field of deep learning for skeleton action recognition.

Keywords: graph convolution; skeletal action recognition; feature fusion

1 Introduction

Skeletal data has drawn considerable attention from numerous researchers due to its high robustness to environmental and lighting variations and relatively low computational demands. Methods for human action recognition based on skeletal data can be broadly categorized into four types: those based on Convolutional Neural Networks (CNNs)[1-2], Recurrent Neural Networks (RNNs)[3-4], Transformers [5-6], and Graph Convolutional Neural Networks (GCNs)[7-12], all of which are continuously evolving. Since skeletal data is non-Euclidean, methods based on graph convolution have a natural advantage in handling such data. Consequently, graph convolution-based approaches are gradually becoming the mainstream method for recognizing skeletal actions.

Yan et al. [7] first proposed a novel skeleton-based action recognition model: ST-GCN. This work significantly shifted attention towards using GCN for skeleton-based action recognition, pioneering the application of graph convolution in skeletal behavior recognition. 2s-AGCN[8] introduced an adaptive graph structure adjustment, allowing for automatic updates using the backpropagation algorithm of neural networks; meanwhile, a dual-stream framework was employed to model first and second-order information and finally perform score fusion, enhancing the model's stability and flexibility. Liu[9] et al. introduced MS-G3D, which constructs a unified spatiotemporal graph containing multiple subgraphs, with each subgraph reflecting the spatial connections of keypoints within a single frame; this form of adjacency matrix effectively models relationships between different keypoints across different frames. Chen[10] et al. developed CTR-GCN, a model that customizes dynamic topological structures for each channel, achieving in-depth aggregation of features from different channels and learning richer topological information. In the same year, Chen[11] and his team proposed MST-GCN, which segments feature maps in the channel dimension and applies graph convolution and temporal convolution separately for feature extraction, then recombines these segments using residual connections, significantly enhancing the model's ability to represent multiscale features. Recently, Zhang[12] et al. proposed SGN, which greatly enhances feature expressiveness by incorporating advanced semantic information of joints (joint type and frame index). Through spatial max-pooling technique, SGN can integrate information from all joints within a single frame, extract key discriminative features, further reducing computational complexity, and achieving model lightweight. Although predecessors have made significant contributions to skeleton-based action recognition using graph convolution, effectively extracting latent information from skeletal data remains a challenge.

2 Related work

2.1 Enhanced Spatial Feature Graph Convolutional Network (ESF-GCN)

The extraction of spatial features is crucial for understanding the spatial complexity of behaviors and body postures. In the backbone model ST-GCN used in this study, the predefined graph structure struggles to adapt to dynamic temporal relationships, limiting the model's ability to comprehend the complex interactions between body parts. Therefore, this study modifies the predefined static graph of GCN in ST-GCN to an adaptive adjacency matrix, updating the coefficient matrix iteratively through gradient descent during training without any sparse constraints. This approach continuously optimizes the graph structure during training, rather than relying on predefined static graphs, allowing the model to better adapt to dynamic changes in joint relationships. Additionally, residual connections are introduced in the GCN part to ensure effective utilization of initial node features. Finally, this study introduces spatial shift operations (Spatial Shift[13]) parallel to GCN, which do not require adjacency matrices but utilize simple graph shifting operations and point-wise convolutions to aggregate neighboring node features, instead of conventional graph convolutions.

In the spatial shift operation, taking a single joint node v_i (left hand) as an example, Figure 1 illustrates the variation between its initial feature vector and the feature vector after spatial shift processing. The black circle in the figure represents the original feature of the central node v_i (left hand). The colored circles represent its offset features, where each original circular feature (channel) corresponds to the same color as the node in its skeletal graph. These offset features are inspired by the non-local shift operation in the Shift-GCN[14]. The large blue arrow on the left indicates subtracting the original feature vector from the shifted feature vector. Arrows of different colors represent the gradient direction of nodes relative to the left hand in the human skeletal graph. Setting x_{0ij} as the original feature of joint node v_i in the j_{th} channel, and x_{ij} as the feature of the same node after the shift operation in the j_{th} channel, derived from the corresponding channel of joint node v_k. Calculating the difference between x_{ij} and x_{0ij} yields the gradient feature of node v_k relative to v_i. By subtracting the original feature map from the shifted feature map, gradient feature maps of all nodes relative to the reference node (such as the left hand) can be obtained.

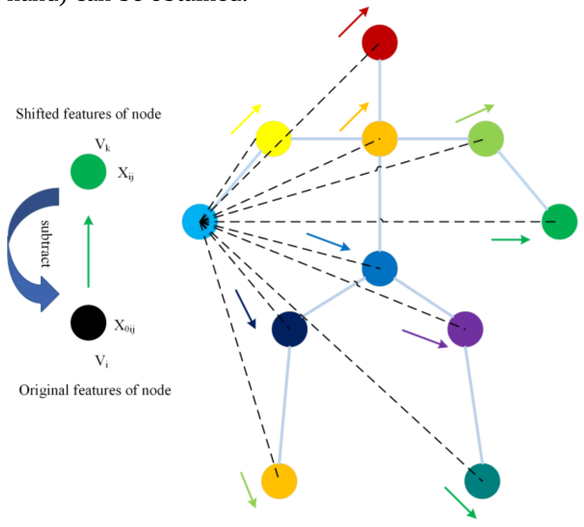

Fig. 1. Spatial Shift operation diagram

In the original spatial skeletal feature map $X_0 \in R^{N \times C}$, where N represents the number of joint nodes and C represents the number of feature channels per node. After spatial shifting, the feature map is represented as $X \in R^{N \times C}$. By performing $X - X_0$, we can extract gradient features of the entire skeletal graph centered around the reference node. Although the connection strength between different joint nodes is assumed to be equal by default, the importance of human body joint nodes varies. Therefore, a trainable mask M is introduced to dynamically adjust the importance of each connection by applying it to the obtained feature map, which can be expressed as follows:

$$Y = W(X - X_0)M \tag{1}$$

In Equation (1), W is a weight matrix composed of weight vectors from multiple output channels, and Y is the output of the spatial shift operation. Subsequently, by introducing the parameter $\alpha \in [0,1]$ to adjust the degree of dependence on the original features during training, where a larger α value indicates greater importance of central difference gradient information. The final expression of the spatial shift module is represented as Equation (2).

$$Y = W(X - \alpha \cdot X_0)M \tag{2}$$

The introduction of spatial shift operation does not lead to a significant increase in parameters and floating point operations. Furthermore, the spatial shift operation enables each node to cover the entire skeletal graph, where the feature of a node after shifting is composed of the features of all other nodes in the graph. Such design enlarges the model's receptive field, allowing it to capture a wider range of spatial relationships. Additionally, important spatial features are extracted through the GCN section. The overall structure of ESF-GCN formed through the fusion of two components is illustrated in the upper part of Figure 2 (a).

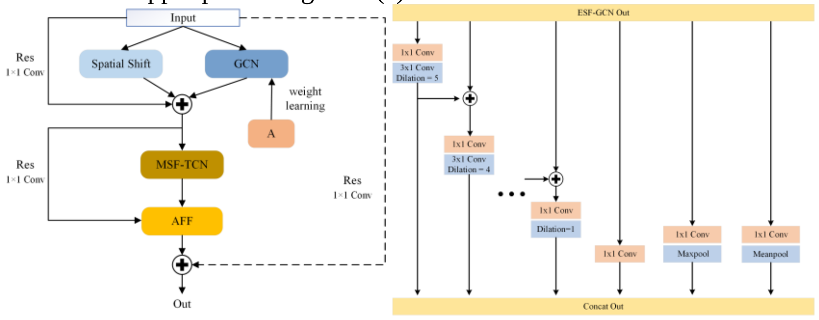

Fig. 2. (a)Basic block of FFN (b) MSF-TCN network architecture diagram

2.2 Multi-Scale Feature Fusion Temporal Convolutional Network (MSF-TCN)

For a comprehensive understanding of behaviors with different durations and complexities, the design of the time feature extraction module is crucial. The backbone network used in this study, ST-GCN, employs one-dimensional convolution on the temporal dimension with a single kernel size of 9 for temporal modeling. The large kernel covers a wide temporal receptive field. While this approach can achieve simple modeling of time, it lacks flexibility, resulting in redundant floating point operations and parameters, and insufficiently detailed feature extraction.

Although researchers[9-10,15] have made some improvements in time convolutional networks, such as multi-scale feature extraction, and have achieved significant progress, further feature exploration and improving model accuracy are still highly necessary. Therefore, this study proposes a Multi-Scale Feature Fusion Temporal Convolutional Network: MSF-TCN.

The structure of MSF-TCN is illustrated in Figure 2(b), consisting of the following components: a "1×1" Conv branch, a Max-Pooling branch, a Mean-Pooling branch, and five 1D Conv branches with kernel sizes of 3 and dilation rates ranging from 1 to 5. The branches are horizontally connected in order of decreasing receptive field size, with representations and their horizontal connections omitted for dilation rates of 3 and 2 in the figure. Initially, the module transforms features using a "1×1" Conv and evenly divides them into eight groups of channel widths. Each feature group is then processed individually. Subsequently, these eight outputs are concatenated together and further processed and outputted by another "1×1" Conv. This new design of temporal convolution not only enhances temporal modeling capabilities but also saves floating point operations and parameters by reducing the channel width of each branch. Additionally, feature supplementation and fusion are achieved through horizontal connections from large to small.

2.3 Attentional Feature Fusion (AFF)

Attention mechanism assigns different weights to different parts of the data to process information. In many previous works[15-17], attention has been incorporated into respective networks to capture the intrinsic topology of human behaviors and specific moments of action occurrence. However, there is still room for improvement in feature extraction with the aforementioned attention mechanisms. Therefore, this paper draws inspiration from reference[18] and employs the Multi-scale Attention Mechanism (MS-CAM), whose structure is depicted in Figure 3 (a). The core idea is to achieve channel attention at multiple scales by altering the size of spatial pooling. This attention is embedded at the output end of MSF-TCN. The input features are fused with global and local attention and then output. Specifically, the structure includes: Local attention constructed through two 1×1 convolution layers (Point-wise Conv), a batch normalization layer, and an activation function (ReLU), mainly capturing local feature information. As for global attention, it first maps features to global information using adaptive average pooling, then further processes global information through two 1×1 convolution layers, a batch normalization layer, and ReLU activation function. In both global and local attention, there are two 1×1 convolution layers, with the first layer reducing the number of channels and the second layer increasing and restoring the number of channels, introducing a parameter 'r' to control the degree of channel compression, defaulting to r=1. The selection of the 'r' parameter needs to balance the relationship between parameter reduction and feature expression capability. Therefore, the overall equation for MS-CAM is represented as Equation (3).

$$Ao = X \otimes M(X) = X \otimes \sigma(L(X) \oplus G(X)) \tag{3}$$

Where $M(X) \in R^{C\times T\times V}$ represents the attention weights generated by MS-CAM, where V represents the number of keypoints, C represents the number of feature channels per node, and T represents the number of frames. In the equation, $\oplus$ denotes broadcast addition, and $\otimes$ denotes element-wise multiplication.

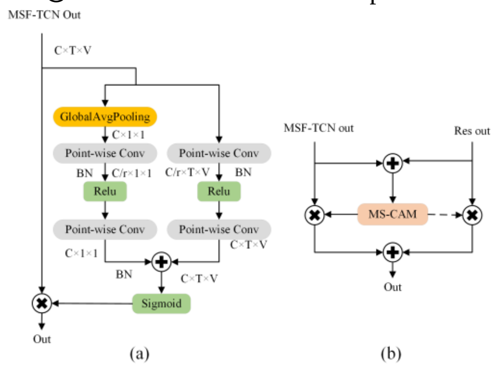

Fig. 3. Schematic diagram of MS-CAM and AFF structures. (a) Schematic diagram of the MS-CAM structure. (b) Schematic diagram of the AFF structure.

However, for the mentioned modules, after undergoing processing by the MSF-TCN module, there is a possibility of losing crucial features, resulting in overfitting. Therefore, Attentional Feature Fusion (AFF) with residual input can be utilized, which consists of two input parts: one part is the output of MSF-TCN, and the other part is the input of MSF-TCN. The multi-input features are then fused with global and local attention and outputted. The calculation process is illustrated in Equation (4).

$$Ao = X \otimes M(X \uplus Y) + Y \otimes (1 - M(X \uplus Y) \tag{4}$$

Where $Ao \in R^{C \times T \times V}$ represents the fused feature, $\uplus$ denotes the initial feature integration. For simplicity, element-wise summation is chosen as the initial integration. The dashed lines in Figure 3 (b) represent $1 - M(X \uplus Y)$. It is worth noting that the fusion weights $M(X \uplus Y)$ consist of real numbers between 0 and 1, and $1 - M(X \uplus Y)$ also consists of real numbers between 0 and 1, allowing the network to perform soft selection or weighted averaging between X and Y.

After the incorporation of AFF, the basic block of FFN in this paper has been designed, as shown in Figure 2. Similar to ST-GCN, FFN consists of 10 basic blocks, with the first basic block lacking the residual connection represented by dashed lines in the figure, while the remaining nine basic blocks all have residual connections.

2.4 Two-stage Loss

Traditional multi-class loss functions, such as cross-entropy loss, often consider all classes equally, which may not be efficient when dealing with class imbalances or samples that are difficult to distinguish. Although some improvements have been proposed by previous researchers[19-21], these enhancements are more evident for datasets with class imbalances, while they still lack sufficient discrimination for datasets with balanced classes.

To address this issue, this study adopts a two-stage training strategy, where the stages are differentiated by different epochs. The first stage is the initial training phase, where the standard cross-entropy loss function is employed as the model's loss function. The goal is to enable the model to capture the basic features and patterns of the data, establishing an initial understanding of the problem. The equation for the multi-class cross-entropy loss function is expressed as follows:

$$L_{CE} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{m} y_{ic} log(p_{ic}) \tag{5}$$

In Equation (5), n represents the number of samples. m is the total number of classes. y_{ic} denotes the true label of sample i for class c. p_{ic} is the probability predicted by the model that sample i belongs to class c.

In the first stage, weighting of high-confidence samples and differentiation of difficult-to-distinguish samples are ignored. Therefore, this study immediately follows the first stage with the design of the second stage. The loss computation in the second stage consists of two parts, one of which is Focal Loss[19], where the parameter γ is used to adjust the weights of positive and negative samples. The equation for this part of computation can be expressed as:

$$L_1 = -(1 - p_{ic})^{\gamma} \log(p_{ic}) \tag{6}$$

Additionally, to weight high-confidence samples, another part of the loss for high-confidence samples is introduced in the second stage, controlled by parameter γ_hc. The computation for this part is represented by equation (7):

$$L_2 = -(1 + p_{ic})^{\gamma_{hc}} \log(p_{ic}) \tag{7}$$

For L_1 and L_2, weighting is performed using a looping factor ξ, which increases as the training epochs progress, gradually strengthening the weight of the L_2 loss. The computation equation is as follows:

$$\xi = \left(f_c \frac{\mathrm{ep}_i}{ep_n} - 1\right) / (f_c - 1) \tag{8}$$

In equation (8), ep_i represents the current training epoch number, ep_n represents the total number of training epochs, and f_c denotes the cycle factor providing adaptiveness for cyclic scheduling. Combining equations (6), (7), and (8), the definition of the loss function for the second stage is as follows:

$$L_{CF} = \xi L_2 + (1 - \xi) L_1 \tag{9}$$

Combining equations (5) and (9) yields the Two-stage Loss equation (10), where the value of m is restricted to $0 \leq m \leq ep_n$.

$$Loss = \begin{cases} L_{CE} & if \quad epoch \leq m \\ L_{CF} & if \quad epoch > m \end{cases} \tag{10}$$

The Two-stage Loss, by integrating the concept of two-stage training, the idea of Focal Loss, and additional attention to high-confidence and difficult-to-distinguish samples, theoretically enables discrimination of difficult-to-distinguish samples while maintaining attention on high-confidence samples.

3 Experiments and Results

3.1 Experimental Setup

The experiment employs 2D and 3D skeleton data from NTU-RGBD 60[22] and NTU-RGBD 120[23] datasets. If 3D data is not labeled, 2D skeleton data is used by default. The experiment follows the skeleton data preprocessing methods provided in the MMAction2 toolbox. The hyperparameters for training are set as follows: the initial learning rate is set to 0.048, and a cosine annealing strategy is utilized to adjust the learning rate. Stochastic Gradient Descent (SGD) with Nesterov momentum of 0.9 is employed to optimize parameters, and weight decay is set to 0.0005. The batch size is set to 54, and the number of training epochs is set to 80. However, when using the Two-stage Loss proposed in this paper as the loss function, the number of training epochs is set to 110.

3.2 Comparison of ESF-GCN Related Experiments

Selection of the α Parameter

This section of the experiment aims to select the optimal parameter α for ESF-GCN. The experiment utilizes ST-GCN as the baseline model, with the GCN component replaced by ESF-GCN, referred to as the S1 model for convenience. This section of the experiment aims to select the optimal parameter α for ESF-GCN. The hyperparameter $\alpha \in [0,1]$ balances the contribution between nodes and gradient information.

The experiment utilizes ST-GCN as the baseline model, with the GCN component replaced by ESF-GCN, referred to as the S1 model for convenience. Based on different α values, comparative training was conducted using the cross-subject protocol on the NTU-RGBD 60 dataset and the cross-setup protocol on the NTU-RGBD-120 dataset, with results shown in Figure 4(a) and (b).

As depicted in Figure 4 (a), when α is set to 0.2, the accuracy is higher compared to other α values, outperforming the baseline ST-GCN model (denoted as S0 dashed line) by 1.74%. Similarly, Figure 4 (b) validates the effectiveness of α set to 0.2, showing an improvement of 2.63% over the ST-GCN model. Therefore, for subsequent experiments related to the ESF-GCN module, α will be set to 0.2. This choice is made because excessively high α values may overly emphasize gradient information, neglecting the importance of original features. Conversely, too low α values may result in the model inadequately utilizing gradient information to adjust feature representations, ultimately reducing the model's ability to learn complex data patterns.

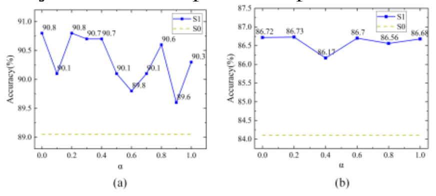

Fig. 4. Comparison of Accuracy between S1 and S0. (a) the cross-subject protocol on NTU RGB+D 60 dataset. (b) the cross-setup protocol on NTU RGB+D 120 dataset.

Comparison between ESF-GCN and GCN

Next, a comprehensive performance comparison between the S1 and S0 models is conducted. As shown in Table 1, compared to S0, S1 achieves a 1.74% and 0.64% improvement on the cross-subject and cross-view protocols of the NTU-RGBD 60 dataset, respectively, without significantly increasing computational costs. Similarly, on the NTU-RGBD 120 dataset, S1 shows improvements of -0.35% and 2.63% on the cross-subject and cross-setup protocols, respectively. These experimental results demonstrate the outstanding performance of ESF-GCN.

Table 1. Comparison of overall performance between S0 and S1.

Model	NTU-RGBD 60		NTU-RGBD 120		*Flops	*Parameters
	X-Sub(%)	X-View(%)	X-Sub(%)	X-Set(%)	(G)	(M)
S0	89.05	96.43	**83.98**	84.1	**3.82**	**3.09**
S1	**90.79**	**97.09**	83.63	**86.73**	4.20	3.41

* $M = 1.0 \times 10^6$，$G = 1.0 \times 10^9$

3.3 Comparison of MSF--TCN Related Experiments

In this section, experiments comparing the MSF-TCN related modules are conducted, with results shown in Table 2. S0-2 represents the model where the TCN in ST-GCN is replaced with MSF-TCN. S2 represents the model where both the GCN and TCN in ST-GCN are replaced with ESF-GCN and MSF-TCN, respectively.

From the data comparison in Table 2, it can be observed that the S0-2 model outperforms the S0 model by 1.68% and 0.56% on the cross-subject and cross-view protocols of the NTU-RGBD 60 dataset, respectively. Similarly, on the NTU-RGBD 120 dataset, the S0-2 model achieves improvements of 0.75% and 3.08% on the cross-subject and cross-setup protocols, respectively, while having only one-third of the parameter count and half of the floating point operations of ST-GCN. These results demonstrate the effectiveness of the MSF-TCN module.

On the other hand, the S2 model exhibits improvements over the S0 model by 2.77% and 0.8% on the cross-subject and cross-view protocols of the NTU-RGBD 60 dataset, respectively. For the NTU-RGBD 120 dataset, the S2 model achieves improvements of 1.21% and 3.21% on the cross-subject and cross-setup protocols, respectively. Compared to the S1 model, the improvements are 1.03%, 0.16%, 1.56%, and 0.58% on different protocols. Moreover, compared to the S0-2 model, the improvements are 1.09%, 0.24%, 0.46%, and 0.13%, respectively. Additionally, both the parameter count and floating point operations of the S2 model are only half of those of the ST-GCN. The experiments demonstrate that integrating two modules into the S2 model simultaneously has not had a negative impact; instead, the performance of the model has been comprehensively improved.

Table 2. Comparison of overall performance between S0,S0-2,S1 and S2.

Model	NTU-RGBD 60		NTU-RGBD 120		*FLOPs	*Parameters
	X-Sub(%)	X-View(%)	X-Sub(%)	X-Set(%)	(G)	(M)
S0	89.05	96.43	83.98	84.1	3.82	3.09
S0-2	90.73	96.99	84.73	87.18	**1.50**	**1.09**
S1	90.79	97.09	83.63	86.73	4.20	3.41
S2	**91.82**	**97.23**	**85.19**	**87.31**	1.88	1.42

3.4 Comparison of AFF Related Experiments

Adding different attention mechanisms to the output part of the S2 model's MSF-TCN forms several new models, as shown in Table 3. Among them, S3(CVSTA) represents the addition of CVSTA[21] to the S2 model. S3(Drop-att) represents the addition of Drop attention[24], both of which are recently effective attention modules. S3(MS-CAM) is the model with MS-CAM added. S3(AFF) represents the model with AFF added, where the subsequent numbers indicate different values of r. These models are trained on the NTU-RGBD 60 dataset using the cross-subject partition method, and the results are obtained as shown in the table.

Table 3 Comparison of accuracy among different attention modules.

Model	Top-1 Accuracy(%)	Top-5 Accuracy (%)
S2	91.82	99.22
S3(CVSTA)	84.10	97.95
S3(Drop-att)	89.36	98.95
S3(MS-CAM)	92.09	99.24
S3(AFF-4)	92.25	99.24
S3(AFF-2)	92.21	99.21
S3(AFF-1)	**92.74**	**99.25**

From Table 3, it can be observed that, except for the attention mechanisms used in this study, the other two attention mechanisms show a decrease in accuracy compared to the original S2 model. Among them, the improvement in S3 (AFF-1) is the most significant, reaching 0.92%. Simultaneously, the experiment also confirms that the optimal value of the r parameter for the AFF module is 1. This may be because when r increases, the features processed by MSF-TCN are already prominent and rich enough. As a result, the increase in r weakens the model's ability to represent features, leading to underfitting of the data that was originally close to fitting.

Comparing the accuracy of S3 (AFF-1) with S2 on the NTU-RGBD 60 dataset using the cross-subject and cross-view partition methods, as well as on the NTU-RGBD 120 dataset using the cross-subject and cross-setup partition methods, the results are shown in Table 4. It can be observed that S3 (AFF-1) outperforms S2 by 0.92%, 0.24%, 0.19%, and 0.72%, respectively, demonstrating the effectiveness of the AFF module.

Table 4 The accuracy comparison results between the S2 and S3(AFF-1) models.

Model	NTU-RGBD 60		NTU-RGBD 120	
	X-Sub(%)	X-View(%)	X-Sub(%)	X-Set(%)
S2	91.82	97.23	85.19	87.31
S3(AFF-1)	**92.74**	**97.47**	**85.38**	**88.03**

Comparing the parameter count and floating point operations of the comprehensive models used in the previous sections, the results are shown in Table 5. Although there is an increase in both parameter count and floating point operations for the S3(AFF-1) model compared to the S2 model, it remains unchanged compared to the baseline model S0. Therefore, considering the significant improvement in accuracy of the S3(AFF-1) model over the S2 model, such increases in parameter count and floating point operations can be considered acceptable.

Table 5 Comparison results of computational and parameter quantities for models with different attention model.

Model	*FLOPs(G)	*Parameters(M)
S0	3.82	3.09
S1	4.20	3.41
S2	**1.88**	**1.42**
S3(CVSTA)	1.94	2.22
S3(Drop-att）	1.88	1.42
S3(MS-CAM)	2.50	2.48
S3(AFF-4)	2.46	2.56
S3(AFF-2)	2.61	2.83
S3(AFF-1)	2.92	3.35

$* M = 1.0 \times 10^6$， $G = 1.0 \times 10^9$

3.5 Comparison of Two-stage Loss Related Experiments

The experiment replaced different loss functions with the cross-entropy loss function used in ST-GCN (S0), and trained on NTU-RGBD 60 using the cross-subject partition method, as shown in Table 6. Poly Loss[20] is an improved version of Focal Loss, which treats the loss function as a linear combination of polynomial functions, making it convenient to adjust the weights. The numbers 2, 6, and 4 represent different values of the cycle factor f_c, which determine different cycle factors ξ accordingly. As for the parameters γ_{hc} and γ, this study adopts the value of 2 for both, based on references[25] and experimental comparisons.

Table 6 Comparison results of experiments with different loss functions

Model	Top-1 Accuracy(%)	Top-5 Accuracy(%)
S0	89.05	98.85
S0(Focal Loss)	89.06	98.94
S0(Poly Loss)	89.32	98.96
S0(Two-stage Loss-2)	90.11	98.99
S0(Two-stage Loss-6)	90.08	99.01
S0(Two-stage Loss-4)	**90.19**	**99.08**

Table 7 Accuracy comparison results between S3 (AFF-1) and S4.

Model	NTU-RGBD 60		NTU-RGBD 120	
	X-Sub(%)	X-View(%)	X-Sub(%)	X-Set(%)
S3(AFF-1)	92.74	97.47	88.03	85.38
S4	**92.79**	**97.46**	**88.26**	**85.48**

From Table 6, it can be observed that when the period factor f_c is set to 4, the accuracy reaches the highest at 90.19%, which is a 1.14% improvement compared to S0, and 1.13% and 0.87% higher than Focal Loss and Poly Loss, respectively.

Subsequently, the cross-entropy loss function in the S3 (AFF-1) model was replaced with Two-stage Loss, with f_c set to 4, resulting in model S4. Training the model on both datasets, Table 7 shows that S4 outperforms S3 (AFF-1) by 0.23% and 0.1% on the cross-subject and cross-setup protocols of the NTU-RGBD 120 dataset, respectively, thus demonstrating the effectiveness of Two-stage Loss.

4.Comparison between FFN and existing models

Comparison between FFN and existing models is presented in this section, as shown in Table 8. "Proposed(j)" and "Proposed(j-3d)" represent models trained with single-stream joint using 2D and 3D keypoint data, respectively. On the other hand, "Proposed(2s)" and "Proposed(4s)" represent models fused with dual-stream and quadruple-stream, respectively, where the fusion ratios are Joint: Bone=1:1 for dual-stream fusion and Joint: Bone: Joint-Motion: Bone-Motion=1:1:0.5:0.5 for quadruple-stream fusion. It can be observed from the comparison that FFN achieves higher accuracy compared to other models, making it a more advanced model. It is worth mentioning that while both ST-GCN in the table and the ST-GCN used in this paper belong to the same model, differences in data processing and training methods can lead to varying results.

Table 8 Comparison Results between FFN and Existing Models

Method	Year	Parameters (M)	NTU-RGBD 60		NTU-RGBD 120	
			X-Sub(%)	X-View(%)	X-Sub(%)	X-Set(%)
VA-LSTM(3d) [26]	ICCV17	-	79.4	87.6	-	-
AGC-LSTM(3d)[27]	CVPR19	22.9	89.2	95.0	-	-
HCN(3d)[28]	IJCAI18	**0.8**	86.5	91.1	-	-
VA-CNN(3d)[26]	TPAMI19	24.09	88.7	94.3	-	-
ST-TR(3d)[6]	ICPR21	12.1	89.9	96.1	82.7	84.7
ST-GCN(2d)[7]	AAAI18	3.08	85.7	92.4	80.1	84.2
2s-AGCN(3d)[8]	CVPR19	3.5	88.5	95.1	82.9	84.9
AAGCN(2d)[16]	CVPR20	3.77	89.7	97.1	80.2	86.3
MS-G3D(2d)[9]	CVPR20	3.17	92.7	97.1	85.5	88.2
CTR-GCN(2d)[10]	ICCV21	1.43	90.6	96.9	82.2	84.5
ST-GCN++(2d)[15]	ACM22	1.39	89.3	97.4	84.4	88.1
PoseC3D(2d)[2]	CVPR22	2.0	94.1	97.1	86.9	90.3
Proposed(j)	-	3.35	92.8	97.5	85.5	88.3
Proposed(j-3d)	-	3.35	88.89	95.55	82.19	84.52
Proposed(2s)	-	3.35	93.72	97.81	86.56	90.08
Proposed(4s)	-	3.35	**94.24**	**98.30**	**87.31**	**90.95**

4 Conclusion

This study is based on the ST-GCN network, which is improved and augmented with ESF-GCN module, MSF-TCN module, AFF module, and Two-stage Loss, and multiple-stream fusion is performed. Through comparison, it is evident that FFN has lower parameter count and higher accuracy, making it a superior model at present.

References

1. Choutas V., Weinzaepfel P., Revaud J., et al. Potion: Pose motion representation for action recognition[C] Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 7024-7033.
2. Duan H., Zhao Y., Chen K., et al. Revisiting skeleton-based action recognition[C]. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022: 2969-2978.
3. Du Y., Wang W., Wang L. Hierarchical recurrent neural network for skeleton based action recognition[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1110-1118.
4. Liu J., Shahroudy A., Xu D., et al. Spatio-temporal lstm with trust gates for 3d human action recognition[C]. Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14. Springer International Publishing, 2016: 816-833.
5. Shi L., Zhang Y., Cheng J., et al. Decoupled spatial-temporal attention network for skeleton-based action-gesture recognition[C]. Proceedings of the Asian conference on computer vision. 2020.
6. Plizzari C., Cannici M., Matteucci M. Spatial temporal transformer network for skeleton-based action recognition[C]. Pattern recognition. ICPR international workshops and challenges: virtual event, January 10–15, 2021, Proceedings, Part III. Springer International Publishing, 2021: 694-701.
7. Yan S., Xiong Y., Lin D. Spatial temporal graph convolutional networks for skeleton-based action recognition[C]. Proceedings of the AAAI conference on artificial intelligence. 2018, 32(1).
8. Shi L., Zhang Y., Cheng J., et al. Two-stream adaptive graph convolutional networks for skeleton-based action recognition[C]. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 12026-12035.
9. Liu Z., Zhang H., Chen Z., et al. Disentangling and unifying graph convolutions for skeleton-based action recognition[C]. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 143-152.
10. Chen Y., Zhang Z., Yuan C., et al. Channel-wise topology refinement graph convolution for skeleton-based action recognition[C]. Proceedings of the IEEE/CVF international conference on computer vision. 2021: 13359-13368.
11. Chen Z., Li S., Yang B., et al. Multi-scale spatial temporal graph convolutional network for skeleton-based action recognition[C]. Proceedings of the AAAI conference on artificial intelligence. 2021, 35(2): 1113-1122.
12. Zhang P., Lan C., Zeng W., et al. Semantics-guided neural networks for efficient skeleton-based human action recognition[C]. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 1112-1121.

13. Miao S., Hou Y., Gao Z., et al. A central difference graph convolutional operator for skeleton-based action recognition[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2021, 32(7): 4893-4899.
14. Cheng K., Zhang Y., He X., et al. Skeleton-based action recognition with shift graph convolutional network[C]. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 183-192.
15. Duan H., Wang J., Chen K., et al. Pyskl: Towards good practices for skeleton action recognition[C]. Proceedings of the 30th ACM International Conference on Multimedia. 2022: 7351-7354.
16. Shi L., Zhang Y., Cheng J., et al. Skeleton-based action recognition with multi-stream adaptive graph convolutional networks[J]. IEEE Transactions on Image Processing, 2020, 29: 9532-9545.
17. Yang H., Ren Z., Yuan H., et al. Multi-scale and attention enhanced graph convolution network for skeleton-based violence action recognition[J]. Frontiers in neurorobotics, 2022, 16: 1091361.
18. Dai Y., Gieseke F., Oehmcke S., et al. Attentional feature fusion[C]. Proceedings of the IEEE/CVF winter conference on applications of computer vision. 2021: 3560-3569.
19. Lin T.Y., Goyal P., Girshick R., et al. Focal loss for dense object detection[C]. Proceedings of the IEEE international conference on computer vision. 2017: 2980-2988.
20. Leng Z., Tan M., Liu C., et al. Polyloss: A polynomial expansion perspective of classification loss functions[J]. ArXiv preprint arXiv:2204.12511, 2022.
21. Liu S.L., Ding Y.N., Zhang J.R., et al. Multi-Dimensional Refinement Graph Convolutional Network with Robust Decouple Loss for Fine-Grained Skeleton-Based Action Recognition[J]. ArXiv preprint arXiv:2306.15321, 2023.
22. Shahroudy A., Liu J., Ng T.T., et al. Ntu rgb+d: A large scale dataset for 3d human activity analysis[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 1010-1019.
23. Liu J., Shahroudy A., Perez M., et al. Ntu rgb+d 120: A large-scale benchmark for 3d human activity understanding[J]. IEEE transactions on pattern analysis and machine intelligence, 2019, 42(10): 2684-2701.
24. Hua Y., Wu W., Zheng C., et al. Part aware contrastive learning for self-supervised action recognition[J]. ArXiv preprint arXiv:2305.00666, 2023.
25. Pang C., Lu X., Lyu L. Skeleton-based action recognition through contrasting two-stream spatial-temporal networks[J]. IEEE Transactions on Multimedia, 2023.
26. Zhang P., Lan C., Xing J., et al. View adaptive neural networks for high performance skeleton-based human action recognition[J]. IEEE transactions on pattern analysis and machine intelligence, 2019, 41(8): 1963-1978.
27. Si C., Chen W., Wang W., et al. An attention enhanced graph convolutional lstm network for skeleton-based action recognition[C]. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 1227-1236.
28. Li C., Zhong Q., Xie D., et al. Co-occurrence feature learning from skeleton data for action recognition and detection with hierarchical aggregation[J]. ArXiv preprint arXiv:1804.06055, 2018.